НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018р.

# ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 “ Комп’ютерні науки ”

на тему: «Моделювання ізотропних поверхонь з квазіконформною заміною параметра»

Виконала: студентка IV курсу, групи ТР-51

Демчук Дарія Вікторівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник старший викладач Гурін Артем Леонідович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

**Національний технічний університет України**
**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “ Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2018р.

# ЗАВДАННЯ
**на дипломну роботу студенту**
Демчук Дарії Вікторівни
(прізвище, ім’я, по батькові)

1. Тема роботи: «Моделювання ізотропних поверхонь з квазіконформною заміною параметра»
керівник роботи Гурін Артем Леонідович, старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201__р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи мова програмування Java, середовище IntelliJ IDEA, бібліотека JZY3D, бібліотека Three.js._____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) розробити алгоритми візуалізації ізотропних поверхонь з квазіконформною заміною параметра, здійснити програмну реалізацію розроблених методів.

5. Перелік ілюстративного матеріалу архітектура системи, графічне представлення інтерфейсу, приклади роботи програмного модулю
6. Дата видачі завдання ” 10 ” жовтня 2018 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітки |
|---|---|---|---|
| 1. | Вивчення та аналіз задачі | 14.10.2018-23.12.2018 | |
| 2. | Розробка архітектури та загальної структури системи | 2.02.2019-3.03.2019 | |
| 3. | Розробка структур окремих підсистем | 4.03.2019-14.04.2019 | |
| 4. | Підготовка матеріалів | 15.04.2019-18.04.2019 | |
| 5. | Програмна реалізація системи | 18.04.2019-14.05.2019 | |
| 6. | Захист програмного продукту | 15.05.2019 | |
| 7. | Оформлення пояснювальної записки | 16.05.2019-3.06.2019 | |
| 8. | Передзахист | 28.05.2019 | |
| 9. | Захист | 17.06.2019-22.06.2019 | |

Студент _____ Демчук Д. В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Гурін А. Л.
(підпис) (прізвище та ініціали,)

# АНОТАЦІЯ

Дипломну роботу виконано на 67 аркушах, вона містить 3 додатки та перелік посилань на використані джерела з 15 найменувань. У роботі наведено 20 рисунків.

Метою роботи було створення системи керування складними геометричними об’єктами побудованими на основі ізотропної геометрії з використанням квазіконформної заміни. Програма забезпечує моделювання ізотропних поверхонь, коефіцієнтів першої та другої квадратичних форм на основі введених значень для розрахунку реперних точок. Розроблений програмний продукт може бути використаний, наприклад, в організаціях та установах, пов’язаних з архітектурною діяльністю.

Ключові слова: Ізотропні поверхні, квазіконформна заміна, моделювання, крива Без’є, квадратична форма, коефіцієнт, реперні точки.

# ABSTRACT

The thesis is presented in 67 pages. It contains 3 appendixes and bibliography of 15 references. Twenty figures are given in the thesis.

The purpose of the work was to create a control system for complex geometric objects constructed on the basis of isotropic geometry using a quasi-conformal replacement. The program provides modeling of isotropic surfaces, coefficients of the first and second quadratic forms on the basis of the entered values for the calculation of reference points. The developed software product can be used, for example, in organizations and institutions related to architectural activities.



Keywords: Isotropic surfaces, quasi-conformal replacement, simulation, Bezier curve, quadratic form, coefficient, reference points.

ЗМІСТ

Перелік умовних скорочень ..... 8

Вступ ..... 9

1 Постановка задачі ..... 10

2 Обчислювальні методи реалізації програмної частини ..... 11

2.1 Перші дослідження мінімальних поверхонь ..... 11

2.2 Аналіз останніх досліджень ..... 14

2.3 Ізотропні поверхні з квазіконформною заміною параметра ..... 19

3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ ..... 23

3.1. Особливості розробки ..... 23

3.2. Обґрунтування використання обраних засобів ..... 23

3.2.1 Мова програмування Java ..... 23

3.2.2 Середовище розробки IntelliJ IDEA ..... 24

3.2.3 Інструмент управління Maven ..... 25

3.2.4 Мова програмування JavaScript і бібліотека ThreeJS ..... 27

3.2.5 Бібліотека Jzy3d ..... 29

3.2.6 Мова розмітки HTML і спеціальна мова стилів CSS ..... 30

3.2.8 Фреймворк Bootstrap ..... 32

3.3 Висновки до розділу ..... 33

4 МЕТОДИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ ..... 34

4.1. Користувацький інтерфейс системи ..... 34

4.2. Засоби зв’язку між клієнтом та сервером ..... 34

4.3. Шаблон архітектури проекту ..... 36

4.5 Архітектура програмного комплексу ..... 37
4.6 Потенційні користувачі ..... 39
4.7 Висновки до розділу ..... 39
5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ..... 40
5.1 Опис web-додатку ..... 40
5.2 Висновки до розділу ..... 48
ВИСНОВКИ ..... 49
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 51
Додаток 1 ..... 53
Додаток 2 ..... 55
Додаток 2 ..... 63

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

| | |
|---|---|
| **MVC** | Model-view-controller — Модель–представлення–контролер |
| **HTML** | Hypertext Markup Language — Мова розмітки гіпертекстових документів |
| **IDE** | Integrated development environment — Інтегроване середовище розробки |
| **CSS** | Cascading Style Sheet — Мова |
| **SOLID** | Single responsibility – Open-closed – Liskov substitution – interface segregation – dependency inversion — Принципи програмування |
| **API** | Application programming interface — Інтерфейс |
| **HTTP** | Hyper Text Transfer Protocol — Протокол |
| **WebGL** | Web-based Graphics Library — Бібліотека |
| **SQL** | Structured query language — Мова структурованих запитів |
| **JSON** | JavaScript Object Notation — Формат файлу |
| **XML** | Extensible Markup Language — Розширена мова розмітки |
| **POM** | Project Object Model — Файл, який описує |
| **SVG** | Scalable Vector Graphics — Специфікація мови розмітки |

# ВСТУП

Важливою підмножиною уявних об’єктів є ізотропні, властивості яких широко застосовують архітектурі та проектуванні. В останні роки набули популярності дослідження на основі ізотропної геометрії. Тому є актуальною проблема моделювання ізотропних поверхонь та створення програмних продуктів, які будуть автоматизувати побудову таких поверхонь. Зазвичай використовують конформну заміну для візуалізації ізотропних поверхонь, але є ряд досліджень, які доводять, що поверхні, для побудови яких використовувалась квазіконформна заміна, є також мінімальними.

Для наочного прикладу було розроблено програмний продукт для моделювання ізотропних поверхонь з квазіконформною заміною параметра, з використанням кривої Без’є, яка застосовувалась в методі побудови як напрямна.

Основними технологіями для розробки такої програми було обрано:

- Java;
- JavaScript;
- Three.js;
- Jzy3d;

Розроблена програма має прямий механізм побудови складних ізотропних об’єктів. Завдяки цьому така система може бути використана в архітектурі для візуалізації траєкторії різьбового профілю, для деяких зводів архітектурних споруд або в розробці медичних приладів.

На основі запропонованих алгоритмів моделювання та побудови поверхонь був розроблений програмний додаток, який по швидкодії перевершує існуючі аналоги.

# 1 ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи було створення системи керування складними геометричними об’єктами побудованими на основі ізотропної геометрії з використанням квазіконформної заміни.

Об’єкт дослідження – комп’ютерні технології ізотропної геометрії.

Предметом дослідження – комп’ютерне моделювання об’єктів на основі ізотропної геометрії з квазіконформною заміною параметра.

Під час роботи над дипломною роботою було визначено ряд завдань для досягнення мети дипломної роботи:

1. проаналізувати основні способи представлення поверхонь на основі ізотропних характеристик;
2. проаналізувати методи моделювання ізотропних поверхонь з квазіконформною заміною параметра;
3. удосконалити метод побудови ізотропних поверхонь з квазіконформною заміною параметра;
4. розробити алгоритми візуалізації ізотропних поверхонь з квазіконформною заміною параметра;
5. здійснити програмну реалізацію розроблених методів, а саме побудова власне ізотропних поверхонь, а також квадратичних форм.

# 2 ОБЧИСЛЮВАЛЬНІ МЕТОДИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ ЧАСТИНИ

При побудові деяких поверхонь постає важливе питання – віднесення площин до конкретних координатних сіток, оскільки тоді вони набувають спеціальних властивостей. Це спрощує нам вирішення задач, які спираються на диференціальні властивості. Ізотропні поверхні мають широкий практичний зміст, оскільки такі поверхні є мінімальними [1].

## 2.1 Перші дослідження мінімальних поверхонь

Перші дослідження мінімальних поверхонь належать Джозефу Луї Лагранжу, який розглянув наступну варіаційну задачу: знайти поверхню з найменшою площею, яка натягнута на даних контур. Лагранж припустив, що шукана поверхня може задаватись у вигляді $z = z(x, y)$. Тоді математик, провівши дослідження, отримав, що функція $z(x, y)$ повинна задовольняти рівнянні Ейлера – Лагранжа [2]:

$$(1 + q^2)\frac{\partial^2 z}{\partial x^2} - 2pq\frac{\partial^2 z}{\partial x \partial y} + (1 + p^2)\frac{\partial^2 z}{\partial y^2} = 0, \qquad (2.1)$$

$$p = \frac{\partial z}{\partial x}\,, q = \frac{\partial z}{\partial y}\,.$$

Пізніше Гаспар Монж, дослідив, що умова мінімальності площі призводить до умови $H = 0$. Саме через цю умову поверхні вважаються «мінімальними». Але насправді потрібно розділяти поняття «мінімальна поверхня» і поверхні з найменшою площею, так як умова мінімальності $H = 0$ – лише необхідна умова мінімальності, яка випливає із першої варіації площі поверхні. Перші методи інтегрування рівняння

Ейлера – Лагранжа були представлені Монжем і Лагранжом у вигляді формул Монжа, які він представив через комплексні характеристики [3]:

$$x = A(t) + A_1(\tau), y = B(t) + B_1(\tau), z = C(t) + C_1(\tau), \quad (2.2)$$

де $t$, $\tau$ – комплексні змінні, $A(t)$ ... $C_1(\tau)$ – голоморфні функції, які задовольняють умову.

$$A'^2(t) + B'^2(t) + C'^2(t) = 0, A_1'^2(\tau) + B_1'^2(\tau) + C_1'^2(\tau) = 0. \quad (2.3)$$

Але ці методи не здобули широкого використання через те, що недостатньо була розвинена теорія функцій комплексного числа. Пуассон лише через декілька років анонсував вирішення задачі Лагранжа – у випадку, коли край поверхні близький до плоскої кривої. Крім мінімальних поверхонь, які були досліджені раніше – катеноїд (рисунок 2.1), гелікоїд (рисунок 2.2), з'явилась ще одна поверхня – Шерка поверхня [4].

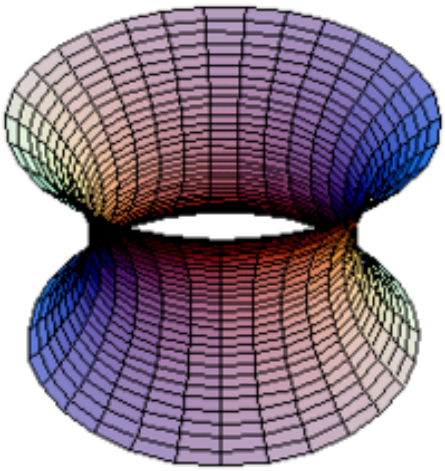

Рисунок 2.1 – Зображення катеноїда

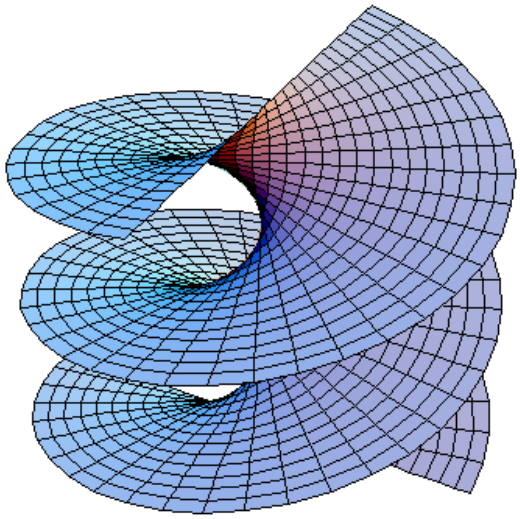

Рисунок 2.2 – Зображення катеноїда

В 1866 році Карл Вейерштрасс вивів формули:

$$\begin{aligned} x &= x_0 + Re \int_0^t (f^2 - g^2)\, dt, \\ y &= y_0 + Re \int_0^t i(f^2 + g^2)\, dt, \\ z &= z_0 + 2Re \int_0^t f g\, dt. \end{aligned} \tag{2.4}$$

Вони являють собою однозв'язну мінімальну поверхню $S(x, y, z)$ і утворюють її через голоморфні функції $f(t), g(t)$ , які визначені в колі або у всій площині зміни внутрішніх ізотермічних координат $(u, v)$, $t = u + vi$. Ці формули (2.4) еквівалентні або містять як часткові випадки всі інші знайдені до цього параметричні представлення мінімальних поверхонь. Вони також дають нам мінімальну поверхню тільки в тому випадку, коли $f$ і $g$ не мають спільних нулів. Якщо $f$ і $g$ мають спільні нулі поверхня є узагальненою з метрикою (2.5) в спільних нулях, яка вироджується [5].

$$ds^2 = (|f|^2 + |g|^2)(du^2 + du^2) \tag{2.5}$$

В цих точках з'являється точки розгалуження мінімальної поверхні. За допомогою формул Вейерштрасса здійснювалась побудова і вивчення багатьох конкретних мінімальних поверхонь, зокрема алгебраїчних мінімальних поверхонь, отриманих в результаті застосування алгебраїчних $f$ і $g$.

В 1874 році Шварц отримав представлення мінімальних поверхонь в ізотерміних координатах $(u, v)$ у вигляді:

$$r(t) = r(u + iv) = Re\left[F(t) - i \int_0^t n(t)dF(t)\right], \tag{2.6}$$

де $F(t)$ і $n(t)$ – тривимірні вектори з голоморфними компонентами, які співпадають відповідно з $r(u, 0)$ і одиничною нормаллю $n(u, 0)$ до мінімальної поверхні.

Праці Вейєрштрасса також привели до широкого використання в теорії мінімальних поверхонь методів і результатів теорії функцій комплексного числа.

## 2.2 Аналіз останніх досліджень

Мінімальні поверхні – це поверхні середня кривизна $H$, якої дорівнює нулю. Умова рівності нулю середньої кривизни є обов’язковою умовою мінімальності площі відсіку поверхні. Це означає, що напруженість у кожній точці мінімальної поверхні є сталою, тому геометрична форма забезпечує рівномірний розподіл зусиль в оболонці та додаткову жорсткість [6].

$$H = \frac{1}{2} \cdot \frac{LG - 2MF + NE}{EG - F^2} = 0, \tag{2.7}$$

де *F, E, G* – коефіцієнти першої квадратичної форми, а *L, M, N* – коефіцієнти другої квадратичної форми.

Перша квадратична форма визначає внутрішню геометрію поверхні в околі даної точки [7]. Вона має вигляд:

$$ds^2 = Edu^2 + 2Fdudv + Gdv^2, \tag{2.8}$$

де $ds$ – лінійний елемент поверхні,

$$\begin{aligned} E &= \left(\frac{\partial X}{\partial u}\right)^2 + \left(\frac{\partial Y}{\partial u}\right)^2 + \left(\frac{\partial Y}{\partial u}\right)^2, \\ F &= \frac{\partial X}{\partial u} \cdot \frac{\partial X}{\partial v} + \frac{\partial Y}{\partial u} \cdot \frac{\partial Y}{\partial v} + \frac{\partial Z}{\partial u} \cdot \frac{\partial Z}{\partial v}, \\ G &= \left(\frac{\partial X}{\partial u}\right)^2 + \left(\frac{\partial Y}{\partial u}\right)^2 + \left(\frac{\partial Y}{\partial u}\right)^2. \end{aligned} \tag{2.9}$$

Друга квадратична форма описує поверхню в другому наближенні. Вона показує, як відхиляється поверхня від дотичної площині і повністю визначає кривизну поверхні [7]. Вона має вигляд:

$$dr \cdot dn = Ldu^2 + 2Mdudv + Ndv^2, \tag{2.10}$$

де $r$ – радіус вектор точки, $n$ – одиничний вектор нормалі,

$$L = \frac{\begin{vmatrix} x'_u & y'_u & z'_u \\ x'_v & y'_v & z'_v \\ x''_{uu} & y''_{uu} & z''_{uu} \end{vmatrix}}{\sqrt{EG - F^2}},$$

$$M = \frac{\begin{vmatrix} x'_u & y'_u & z'_u \\ x'_v & y'_v & z'_v \\ x''_{uv} & y''_{uv} & z''_{uv} \end{vmatrix}}{\sqrt{EG - F^2}}, \qquad (2.11)$$

$$N = \frac{\begin{vmatrix} x'_u & y'_u & z'_u \\ x'_v & y'_v & z'_v \\ x''_{vv} & y''_{vv} & z''_{vv} \end{vmatrix}}{\sqrt{EG - F^2}}.$$

Розглянемо один із способів побудови мінімальних поверхонь. Нехай задана деяка ізотропна плоска крива: $r(t)$. Побудова будь яких ізотропних кривих будемо проводити за допомогою визначення дійсних ознак точок та визначення уявних:

$$r_{j\,Re} \Rightarrow r_{j\,Im}. \qquad (2.12)$$

Для побудови сітки будемо використовувати ізотропну криву, замість параметру $t$ підставимо деяку комплексну змінну.

Будемо називати конформною заміною, якщо замість параметру $t$ підставимо комплексну замінну (2.13)

$$u + iv : t = u + iv. \qquad (2.13)$$

Будемо називати квазіконформною заміною, якщо замість параметру $t$ підставимо комплексну замінну (2.14)

$$u + ikv : t = u + ikv, \text{ або } ku + iv : t = ku + iv. \qquad (2.14)$$

В результаті підстановок отримаємо функцію: $r = r(u, v)$. Якщо відокремлювати дійсну частину від уявної та відображувати на площині, то отримаємо дві сітки (2.15) :

$$r_{Re}(u, v) = Re\big(r(u, v)\big) \text{ та } r_{Im}(u, v) = Im(r(u, v)). \tag{2.15}$$

Напрямними кривими в цих сітках будуть дійсна та уявна частини ізотропної кривої. Властивості отриманих сіток будуть залежати від способу побудови напрямної кривої та заміни параметру.

Поверхню в комплексному просторі визначаємо кінематичним способом за допомогою руху однією ізотропної кривої (твірної) за іншими заданими ізотропними кривими (напрямними). За допомогою виділення окремо дійсної та уявної частини будемо мати дві поверхні для дослідження (2.16):

$$\begin{gathered} x_{Re}(u, v) = Re\big(x(u, v)\big), \qquad y_{Re}(u, v) = Re\big(y(u, v)\big), \\ z_{Re}(u, v) = Re\big(z(u, v)\big); \end{gathered} \tag{2.16}$$

$$\begin{gathered} x_{Im}(u, v) = Im\big(x(u, v)\big), \qquad y_{Im}(u, v) = Im\big(y(u, v)\big), \\ z_{Im}(u, v) = Im\big(z(u, v)\big); \end{gathered}$$

При застосуванні методу Вейєрштрасса, поверхню будемо будувати на основі напрямної ізотропної кривої, конформної заміни, або квазіконформної заміни параметру (2.17) :

$$\begin{gathered} x_{Re}(u, v) = Re\big(x(t)\big), \qquad y_{Re}(u, v) = Re\big(y(t)\big), \\ z_{Re}(u, v) = Re\big(z(t)\big); \end{gathered} \tag{2.17}$$

$$x_{Im}(u,v) = Im\big(x(t)\big), \qquad y_{Im}(u,v) = Im\big(y(t)\big),$$
$$z_{Im}(u,v) = Im\big(z(t)\big),$$

де $x = x(t)$, $y = y(t)$, $z = z(t)$ - просторова параметрична ізотропна крива.

При конформній і квазіконформній заміні параметру виділення дійсної та уявної частини дозволяє одержувати мінімальні та приєднані мінімальні поверхні у дійсному просторі. Якщо в рівнянні поверхні вирази, де $x = x(t)$, $y = y(t)$, $z = z(t)$ визначають плоску ізотропну криву, тоді будемо мати ортогональну сітку на площині [8]. Алгоритм формування об'єкту дійсного простору (рисунок 2.3)


Рисунок 2.3 – Алгоритм побудови об'єкту дійсного простору на основі ізотропної геометрії

## 2.3 Ізотропні поверхні з квазіконформною заміною параметра

Для моделювання поверхні застосуємо метод Вейєрштрасса [2], тобто поверхню будемо будувати на основі напрямної ізотропної кривої, але замість конформної заміни параметра t = u + vi будемо використовувати квазіконформну заміну: t=ku+iv або t=u+ikv. Рівняння поверхні будемо одержувати на основі виділення дійсної або уявної частини від отриманого виразу:

$$\begin{aligned} &x_{Re}(u,v) = Re\big(x(t)\big),\ \ y_{Re}(u,v) = Re\big(y(t)\big),\ \ \ z_{Re}\,(u,v) = Re\big(z(t)\big) \\ &x_{Im}(u,v) = Im\big(x(t)\big),\ \ y_{Im}(u,v) = Im\big(y(t)\big),\ \ \ z_{Im}\,(u,v) = Im\big(z(t)\big), \end{aligned} \tag{2.18}$$

де $x = x(t)$, $y = y(t)$, $z = z(t)$ - просторова параметрична ізотропна крива.

Візьмемо в якості напрямної кривої - криву Без'є у вигляді:

$$r(t) = \sum_{j=0}^{n} r_j J_{n,j}(t)\ \text{, де}\ J_{n,j}(t) = \frac{n!}{j!(n-j)!}\,t^j(1-t)^{n-j}\ \text{, де}\ \mathrm{r_j} = [x_j\ \ y_j\,]. \tag{2.19}$$

Розглянемо частковий випадок, а саме підставимо у рівняння (2.11) n = 3, тобто в якості напрямної будемо застосовувати кубічну ізотропну криву Без'є. Виконаємо заміну t=u+ikv та відокремимо дійсну частину, одержимо:

$$
\begin{aligned}
\mathrm{r}_{Re}(\mathrm{u},\mathrm{v}) = {} & r_{0Re}(1-3u+3u^2-3v^2-u^3+3uv^2k^2) \\
& - r_{0Im}(-3vk+6vku-3u^2vk+v^3k^3) \\
& - \big(-3r_{1Re}(1-2u+u^2-v^2k^2)+3r_{1Im}(-2vk+2uvk)\big)u \\
& + (-3r_{1Im}\big((1-2u+u^2-v^2k^2)-3r_{1Re}(-2vk+2uvk)\big)vk \\
& - (-3r_{2Re}(1-u)-3r_{2Im}vk)(u^2-v^2k^2) \\
& + 2(-3r_{2Im}(1-u)+3r_{2Re}vk)vku + r_{3Re}(u^3-3uv^2k^2) \\
& - r_{3Im}(u^2-vk-v^3k^3).
\end{aligned}
\tag{2.20}
$$

При заміні t = ku+iv будемо мати:

$$
\begin{aligned}
r_{Re}(u,v) = {} & r_{0Re}(1-3uk+3u^2k^2-3v^2-u^3k^3+3uv^2k) \\
& - r_{0Im}(-3v+6uvk-3u^2vk^2+v^3) \\
& - \big(-3r_{1Re}(1-2uk+u^2k^2-v^2)+3r_{1Im}(-2v+2uvk)\big)uk \\
& + \big(-3r_{1Im}(1-2uk+u^2k^2-v^2)-3r_{1Re}(-2v+2uvk)\big)v \\
& - (-3r_{2Re}(1-uk)-3r_{2Im}v)(u^2k^2-v^2)+2(-3r_{2Im}(1-uk) \\
& + 3r_{3Re}(u^3v^3-3uv^2k) - r_{3Im}(3u^2k^2v-v^3).
\end{aligned}
\tag{2.21}
$$

Нехай напрямна ізотропна крива Без'є створюється на основі аналітичної функції:

$$r(t) = r_0(1-t)^3 + 3r_1(1-t)^2t + 3r_2(1-t)t^2 + r_3t^3, \text{ де} \tag{2.22}$$

$r_0[(a_0-a_2)i \quad a_0+a_2 \quad -a_1i]$,

$$
\begin{aligned}
& r_1\big[(a_0-a_2-a_3)i \quad a_0+a_2+a_3 \quad (a_3-a_{1)}i\big], \\
& r_2\big[(a_0-a_2-2a_3)i \quad a_0+a_2+2a_3 \quad (a_3-a_{1)}i\big], \\
& r_3\big[(a_0-a_2-2a_3)i \quad a_0+a_2+4a_3 \quad (3a_3-a_{1)}i\big].
\end{aligned}
\tag{2.23}
$$

Підставимо значення реперних точок (2.23) у рівняння (2.20) та (2.21). Розрахуємо коефіцієнти першої та другої квадратичних форм.

Для *t=u+ikv* коефіцієнти першої квадратичної форми:

$$E = 9a_{3Re}^2[2u^2v^2k^2 + v^4k^4 + 2v^2k^2 + 1 + 2u^2 + u^4] + 9a_{3Im}^2[v^4k^4 + 2v^2k^2 + 2v^2k^2u^2 + 1 + 2u^2 + u^4], \quad (2.24)$$

$$G = 9a_{3Re}^2[2u^2v^2k^4 + u^4k^2 + 2u^2k^2 + v^4k^4 + 2v^2k^4 + k^2] + 9a_{3Im}^2[2u^2v^2k^4 + v^4k^6 + 2v^2k^4 + u^4k^2 + 2u^2k^2 + k^2],$$

$$F = 0.$$

Аналіз виразів (2.16) показує, що $E \neq G \Rightarrow k^2 E = G$.

Розрахуємо коефіцієнти другої квадратичної форми:

$$L = \frac{6a_{3Im}k(a_{3Re}^2[2u^2k^2v^2 + v^4k^4 + 2v^2k^2 + 1 + 2u^2 + u^4]}{\sqrt{(a_{3Im}^2 + (a_{3Re}^2)^2(v^2k^2 + 1 + u^2)^4k^2}} + \frac{a_{3Im}^2[v^4k^4 + 2v^2k^2 + 2u^2k^2v^2 + 1 + 2u^2 + u^4])}{\sqrt{(a_{3Im}^2 + (a_{3Re}^2)^2(v^2k^2 + 1 + u^2)^4k^2}} \quad (2.25)$$

$$M = \frac{6a_{3Re}k^2(a_{3Re}^2[2u^2k^2v^2 + v^4k^4 + 2v^2k^2 + 1 + 2u^2 + u^4]}{\sqrt{(a_{3Im}^2 + (a_{3Re}^2)^2(v^2k^2 + 1 + u^2)^4k^2}} + \frac{a_{3Im}^2[v^4k^4 + 2v^2k^2 + 2u^2k^2v^2 + 1 + 2u^2 + u^4])}{\sqrt{(a_{3Im}^2 + (a_{3Re}^2)^2(v^2k^2 + 1 + u^2)^4k^2}}$$

$$N = -\frac{6a_{3Re}k^3(a_{3Re}^2[2u^2k^2v^2 + v^4k^4 + 2v^2k^2 + 1 + 2u^2 + u^4]}{\sqrt{(a_{3Im}^2 + (a_{3Re}^2)^2(v^2k^2 + 1 + u^2)^4k^2}} + \frac{a_{3Im}^2[v^4k^4 + 2v^2k^2 + 2u^2k^2v^2 + 1 + 2u^2 + u^4])}{\sqrt{(a_{3Im}^2 + (a_{3Re}^2)^2(v^2k^2 + 1 + u^2)^4k^2}}$$

Аналізуючи вирази (2.17), отримаємо наступні співвідношення:

$$k^2L = -N\ ,\ M = \frac{a_{3Re}Lk}{a_{3Im}}. \tag{2.26}$$

Знайдемо середню кривину поверхні. Для цього проаналізуємо вираз чисельника:

$$LG = -\frac{N}{k^2}Ek^2 = -NE,\ \text{MF} = \text{M} \cdot 0 = 0; \tag{2.27}$$

На основі виразу (2.26) будемо мати *H=0*, тобто поверхня буде мінімальною.

*Я*кщо провести аналогічні дослідження з квазіконформною заміною *t=ku+iv*, то одержимо нульове значення середньої кривини, тобто поверхня буде мінімальною [9].

## 2.4. Висновки до розділу

В даному розділі ми розглянули історію дослідження мінімальних поверхонь, методи моделювання, квадратичні форми поверхонь. Вимогою мінімальності поверхні є рівність нулю. Для побудови ізотропної поверхні ми використовували метод напрямної кривої – виділяли дійсні та уявні ознаки точок. Розглянули частковий випадок – використання в якості напрямної кривої криву Без'є третього порядку, також використання квазіконформної заміни, за допомогою якої ми все ж отримали очікуваний результат – кривизна поверхні дорівнювала 0, отже, поверхня мінімальна.

# 3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

## 3.1. Особливості розробки

Розроблений web-додаток для побудови складних поверхонь складається з клієнтської частини (HTML з використанням JavaScript, Bootstrap, CSS) та серверної частини (мови програмування Java). Клієнтська частина, що розроблена засобами JavaScript, призначена для візуалізації вводу даних та відправлення їх на сервер для перерахунків поверхонь. У свою чергу, серверна частина використовується для обрахунків поверхонь, квадратичних форм поверхні та візуалізації поверхонь завдяки інструментам бібліотеки Jzy3D. Середовище розробки програмного продукту використовувалась найоптимальніша – IntelliJ IDEA. Інструмент для керування проектом – Maven.

## 3.2. Обґрунтування використання обраних засобів

### 3.2.1 Мова програмування Java

Мова програмування Java – це мова високого рівня, що нагадує мову людини. На відміну від мов низького рівня, які нагадують машинний код, мови високого рівня повинні бути перетворені за допомогою компіляторів або інтерпретаторів. Це спрощує розробку, роблячи мову легшою для написання, читання і підтримування. Інша характеристика, яка стала причиною вибору мови Java для розробки додатку, є

те, що це мова об'єктно-орієнтованого програмування. Розробка додатків ООП набагато простіше, а також допомагає підтримувати модульну, гнучку та розширювану систему.

Знання про ключові поняття ООП, такі як абстракція, інкапсуляція, поліморфізм і успадкування, дає можливість користуватись всіма перевагами, які дає нам Java. Сама мова втілює багато кращих практик і шаблон дизайну в його бібліотеці. Java також сприяє використанню SOLID і Object-орієнтованих принципів проектування у вигляді проектів з відкритим вихідним кодом. Сама мова Java дуже проста. Проте Java поставляється з бібліотеки класів, яка надає найчастіше використовувані функції утиліти, без яких більшість програм Java не обійтися. Ця бібліотека класів, яка називається Java API, є частиною Java як самої мови. Мова Java має лише 50 ключових слів, але Java API має кілька тисяч класів - з десятками тисяч методів, які можна використовувати у ваших програмах. Вам не потрібно вивчати все API Java. Більшість програмістів володіють лише незначною його частиною. Якщо вам потрібно використовувати якийсь клас з API, з яким ви ще не знайомі, ви можете переглянути те, що робить клас у документації Java API [10]. Розробка на цій мові дає нам можливість використовувати потужне середовище розробки таке як IntelliJ IDEA.

### 3.2.2 Середовище розробки IntelliJ IDEA

Середовище розробки IntelliJ IDEA - це інтегрована середовище розробки Java (IDE), яке користується великою популярністю серед розробників програмного забезпечення та компаній-виробників програмного забезпечення. Додаток завантажується з дивовижним набором функцій та інструментів, які чудово підходять для розробки програмного забезпечення. IntelliJ IDEA розроблена з метою

підвищення продуктивності шляхом надання найбільш інтуїтивної кодової допомоги для всіх підтримуваних мов і фреймворків. Середовище розробки IntelliJ IDEA надає функції налагодження для більшості додатків, які можуть підтримувати такі функції, як перегляд і маніпулювання об'єктами під час запуску коду. IDEA повністю підтримує розробку додатків Android. Необхідна мінімальна установка для того, щоб IDE було готове до використання. Великою перевагою є можливість зміни структури папок проекту на основі налаштувань, наприклад, ви можете змінити структуру з перегляду пакунками програми, а не файлами проекту [11]. Характеристики середовища розробки IntelliJ IDEA:

- інструменти побудови;
- контроль версій;
- розумне завершення;
- аналіз потоку даних;
- ін'єкція мови;
- послідовне перетворення;
- виявлення дублікатів;
- швидкий пошук;
- евристика;
- декомпілятор;
- інструменти бази даних / SQL;
- відладчик;
- контроль версій;
- Gradle та ін.

Середовище розробки IntelliJ IDEA дозволило нам використовувати Maven для створення нашого веб додатку.

### 3.2.3 Інструмент управління Maven

Інструмент управління Maven - це інструмент управління проектами, який надає розробникам повну структуру життєвого циклу. Команда розробників може автоматизувати побудову інфраструктури проекту практично в будь-який час, оскільки Maven використовує стандартну розкладку каталогів і життєвий цикл побудови за замовчуванням. У випадку декількох середовищ розробки команд, Maven може налаштувати спосіб роботи відповідно до стандартів за дуже короткий час. Оскільки більшість установок проекту є простими і багаторазовими, Maven робить життя розробника легким під час створення звітів, перевірок, побудови та тестування автоматизації.

Інструмент управління Maven надає розробникам можливості:

- просте налаштування проекту, що відповідає найкращим практикам;
- послідовне використання всіх проектів;
- Управління залежностями, включаючи автоматичне оновлення.
- Велике і зростаюче сховище бібліотек.
- Легко писати плагіни на мовах Java або сценаріїв.
- Миттєвий доступ до нових функцій з невеликою або без додаткової конфігурації.

Інструмент Maven спрощує і стандартизує процес створення проекту. Він обробляє компіляцію, розповсюдження, документацію, командну співпрацю та інші завдання. Maven підвищує можливості повторного використання і піклується про більшість завдань, пов'язаних з побудовою [12].

Основна мета Maven – надати розробнику наступні можливості:

1. Комплексна модель для проектів, яка є багаторазовою, доступною для налагодження та легшою для розуміння.

2. Додатки або інструменти, які взаємодіють з цією декларативною моделлю.

Структура і вміст проекту Maven оголошуються у файлі xml, який називається Project Object Model (POM), і який є основною одиницею всієї системи Maven. Це дозволило записати потрібні залежності до цього файлу, і не потрібно шукати бібліотеки з інтернету.

### 3.2.4 Мова програмування JavaScript і бібліотека ThreeJS

Фреймворк Three.js – це найпопулярніший у світі фреймворк JavaScript для відображення 3D-вмісту в Інтернеті, який надає вам можливість відображати неймовірні моделі, ігри, музичні відео, наукові та візуалізації даних, або майже все, що можна собі уявити браузера та на смартфоні. З його допомогою можна створювати камери, об'єкти, підсвічування, матеріали та багато іншого, і у вас є вибір візуалізації – це означає, що ви можете вирішити, чи потрібно, щоб ваша сцена була намальована за допомогою полотна HTML 5, WebGL або SVG. А оскільки це відкрите джерело, ви можете навіть долучитися до проекту.

Функціональні можливості Three.js [13]:

- ефекти: Анагліф, перехресний і паралаксний бар'єр;
- сцени: додавання та видалення об'єктів під час виконання; туман;

- камери: перспективні та орфографічні; контролери: трекбол, FPS, шлях і багато іншого;
- анімація: арматура, кінематика вперед, зворотна кінематика, морф і ключовий кадр;
- світильники: освітлення навколишнього середовища, напрямок, точка і світло; тіні: відливання і отримання;
- матеріали: Ламберт, Фонг, гладке затінення, текстури тощо;
- шейдери: доступ до повних можливостей OpenGL Shading Language (GLSL): відблиск об'єктивів, глибина пропуску та обширна бібліотека пост-обробки;
- об'єкти: сітки, частинки, спрайти, лінії, стрічки, кістки та багато іншого - все з рівнем деталізації;
- геометрія: площина, куб, сфера, тор, 3D текст тощо; модифікатори: токарний, екструзійний і трубний;
- завантажувачі даних: двійкові, зображення, JSON і сцена;
- утиліти: повний набір часових та 3D-математичних функцій, включаючи frustum, matrix, quaternion, UVs та багато іншого;
- експортувати та імпортувати: утиліти для створення файлів JSON-сумісних файлів Three.js: Blender, openCTM, FBX, Max і OBJ;
- підтримка: Документація API знаходиться в стадії розробки, публічний форум і вікі в повному обсязі;
- приклади: більше 150 файлів прикладів кодування плюс шрифти, моделі, текстури, звуки та інші файли підтримки;
- налагодження: Stats.js, [9] Інспектор WebGL, [10] Інспектор Three.js [11];
- віртуальна реальність: доступ до WebVR [12];

За допомогою цього фреймворку ми мали змогу створити приклади для перегляду і маніпуляцій ізотропною поверхнею.

### 3.2.5 Бібліотека Jzy3d

На даному етапі використання бібліотеки Jzy3d надає робочі 3d макети і показує, як їх виконувати у простому графіку Jzy3d. Графіки можуть бути покращені наступним чином:

Окремі кольорові краї ускладнюють розуміння обсягу графіка. Кластери або шляхи фарбування надають візуальні орієнтири, що допомагають візуалізувати структуру графіка.

Мітка вузла реалізована, але заплутана графіками. Підказки, які чутливі до миші, можуть збільшити читабельність графіків.

В бібліотеці Jzy3d масштабування і панорамування моделі працюють наступним чином: повертаючись навколо форми, і змінюючи його Z масштабування з колесом миші або клавіатури.

Кешування результатів макета дозволить вам отримати багато часу на розробку графічної частини програми.

Можуть бути й інші цікаві підходи до використання третього виміру:

Складання декількох 2d-графіків із загальним макетом, щоб побачити, як властивості вузла та краю (розмір, колір) розвиваються з плином часу [14].

Нехай (X, Y) фіксується існуючим алгоритмом 2d і обчислює Z відповідно до значення атрибута вузла. Зміна вигляду верхнього 2d у вигляд профілю негайно відобразить категорії вузлів відповідно до їх значення атрибута.

### 3.2.6 Мова розмітки HTML і спеціальна мова стилів CSS

Мова розмітки HTML або Hyper-text-Markup-Language - це загальноприйнята мова програмування для форматування веб-сторінок. У сучасному світі, це зазвичай використовується разом з JavaScript і каскадні таблиці стилів (CSS), щоб дати веб-сторінкам вигляд, який ми хочемо. За допомогою HTML можна змінити формат і зовнішній вигляд зображень, посилань, заголовків, тексту, макета сторінки і майже кожного елемента веб-сторінки. Хоча сьогодні існують інші глобально визнані мови та засоби веб-програмування, такі як системи управління контентом, HTML продовжує залишатися найважливішою мовою програмування для створення веб-сторінок. Вона також є найбільш оптимальним для більшості малих і зростаючих підприємств, які не потребують розширеної функціональності на своєму веб-сайті. Створення сайту за допомогою HTML досить просте. Більше того, вам не потрібно турбуватися про прийняття кількох заходів для індексування вашого сайту. Незважаючи на те, що веб-сайт здатний добре класифікувати в Інтернеті, ви можете легко зробити необхідні налаштування для підвищення SEO оптимізації. Сканери пошукової системи можуть легко ідентифікувати ваш сайт, якщо коди чисті та не містять помилок. Таким чином, сканери не займуть багато часу при індексації вмісту веб-сторінок, і ваш веб-сайт може бути легко виявлений.

*Каскадні таблиці стилів*, що називаються CSS, є простою мовою дизайну, призначеної для спрощення процесу створення веб-сторінок. CSS обробляє зовнішній вигляд веб-сторінки. Використовуючи CSS, можна керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення або кольори використовуються, макетом, варіантами відображення для різних пристроїв і розмірів екрана а також безліч інших ефектів.

Мова CSS легко вчитися і розуміти, але він забезпечує потужний контроль над презентацією HTML-документа. Найчастіше, CSS поєднується з мовами розмітки HTML або XHTML.

Переваги CSS:

*Спеціальна мова стилів CSS економить час* – можна писати CSS один раз, а потім повторно використовувати той самий аркуш на декількох сторінках HTML. Ви можете визначити стиль для кожного елемента HTML і застосувати його до якомога більшої кількості веб-сторінок.

*Сторінки завантажуються швидше*. Якщо ви використовуєте CSS, вам не потрібно кожного разу писати атрибути тегів HTML. Просто напишіть одне правило CSS тегу і застосуйте його до всіх входжень цього тегу. Тому менше коду означає швидше завантаження.

*Легке технічне обслуговування* – Щоб зробити глобальні зміни, просто змініть стиль, і всі елементи на всіх веб-сторінках будуть автоматично оновлюватися.

*Чудові стилі для HTML* – CSS має набагато ширший масив атрибутів, ніж HTML, тому ви можете набагато краще поглянути на вашу HTML-сторінку порівняно з атрибутами HTML.

*Сумісність з кількома пристроями* – таблиці стилів дозволяють оптимізувати вміст для більш ніж одного типу пристроїв. Використовуючи один і той же документ HTML, для портативних пристроїв, таких як КПК і стільникові телефони або для друку, можуть бути представлені різні версії веб-сайту.

*Глобальні веб-стандарти* – тепер атрибути HTML застаріли, і рекомендується використовувати CSS. Таким чином, це гарна ідея, щоб почати використовувати CSS на всіх HTML-сторінках, щоб зробити їх сумісними з майбутніми браузерами [15].

### 3.2.8 Фреймворк Bootstrap

Фреймворк Bootstrap – це безкоштовний і відкритий набір інструментів для створення веб-сайтів і веб-додатків. Це найпопулярніший HTML, CSS і JavaScript фреймворк для розробки адаптивних веб-сайтів для мобільних пристроїв.

З бібліотекою Bootstrap ви можете зробити багато речей, які пришвидшують розробку:

- Ви можете легко створювати відповідні веб-сайти.
- Можна швидко створити багатоколоновий макет з попередньо визначеними класами.
- Ви можете швидко створювати різні типи макетів форм.
- Можна швидко створити різні варіанти навігаційної панелі.
- Ви можете легко створювати такі компоненти, як акордеони, модальні пристрої тощо без написання будь-якого JS-коду.
- Можна легко створювати динамічні вкладки для керування великим вмістом.
- Ви можете легко створювати підказки та вікна, що показують текст підказки.
- Ви можете легко створити карусель або повзунок зображення, щоб продемонструвати свій вміст.
- Ви можете швидко створити різні типи блоків оповіщення.

Переваги використання Bootstrap:

*Заощаджуйте багато часу* – Ви можете заощадити багато часу і зусиль, використовуючи попередньо визначені шаблони дизайну і класи Bootstrap і зосередитися на інших роботах з розробки.

*Особливості, що реагують* – за допомогою Bootstrap можна легко створювати відповідні веб-сайти, які більш відповідним чином відображаються на різних пристроях і дозволах екрана без будь-яких змін у розмітці.

*Послідовний дизайн* – усі компоненти Bootstrap мають спільні шаблони та стилі дизайну через центральну бібліотеку, тому дизайн і розташування веб-сторінок будуть послідовними.

*Проста у використанні* – Bootstrap дуже проста у використанні. Будь-хто з базовими знаннями HTML, CSS і JavaScript може розпочати розробку за допомогою Bootstrap.

*Сумісність із браузерами* – Bootstrap створюється з урахуванням сучасних веб-браузерів і сумісний з усіма сучасними браузерами, такими як Chrome, Firefox, Safari, Internet Explorer тощо.

*Відкритий доступ до коду (Open Source)* – І найкраща частина, це абсолютно безкоштовно завантажити і використовувати [16].

## 3.3 Висновки до розділу

Проаналізувавши плюси та мінуси сучасних мов програмування та засобів розробки, було вирішено використовувати такі, які були представлені вище, оскільки вони мають переваги над іншими для створення веб додатку, зокрема, в часі роботи над створенням, що дозволило в короткі терміни створити повноцінну програму для використання.

# 4 МЕТОДИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

## 4.1. Користувацький інтерфейс системи

Дизайн користувацького інтерфейсу створений на основі трьох принципів [17]:

- Контроль користувачем інтерфейсу;
- Зменшення навантаження пам’яті користувача;
- Послідовність інтерфейсу.

Розроблений веб додаток має інтуїтивно простий інтерфейс, що дозволяє легко користуватись сайтом. Мінімальна кількість елементів перешкоджає навантаженню пам’яті. Послідовність інтерфейсу полягає у правильності розміщення кнопок головного меню, а також самих побудов. Користувачеві дозволяється змінювати мову для зручності навігації по сайту. Після побудови поверхні, користувач може здійснювати анімації, масштабування, що забезпечує краще дослідження отриманого результату.

## 4.2. Засоби зв’язку між клієнтом та сервером

Клієнт та сервер взаємодіють за допомогою HTTP-запитів. Клієнт у нашому випадку (рисунок 3.1) – браузер. Вій здійснює запит, який отримує web-контейнер. В якості такого контейнера ми використовували TomCat – контейнер сервлетів. Коли отримуємо запит, здійснюється парсинг HTTP-запиту, перетворюємо його в деяку інструкцію, яка повинна бути виконана в контексті даного веб додатку. Отримавши

вказівки, контейнер передає управління веб компонентам. Це можуть бути слухачі, фільтри, а також сервлети. В даному додатку ми спочатку потрапляємо у фільтр локалізації для налаштування мови відображення веб застосунку. Тоді управління переходить до сервлету. Виконавши інструкцію, результат повертається у веб контейнер, він перетворює його у HTTP-відповідь. Важлива перевага використання контейнеру – це те, що нам не доводиться винаходити свій парсер для протоколів, що значно пришвидшує розробку будь яких веб додатків [18].

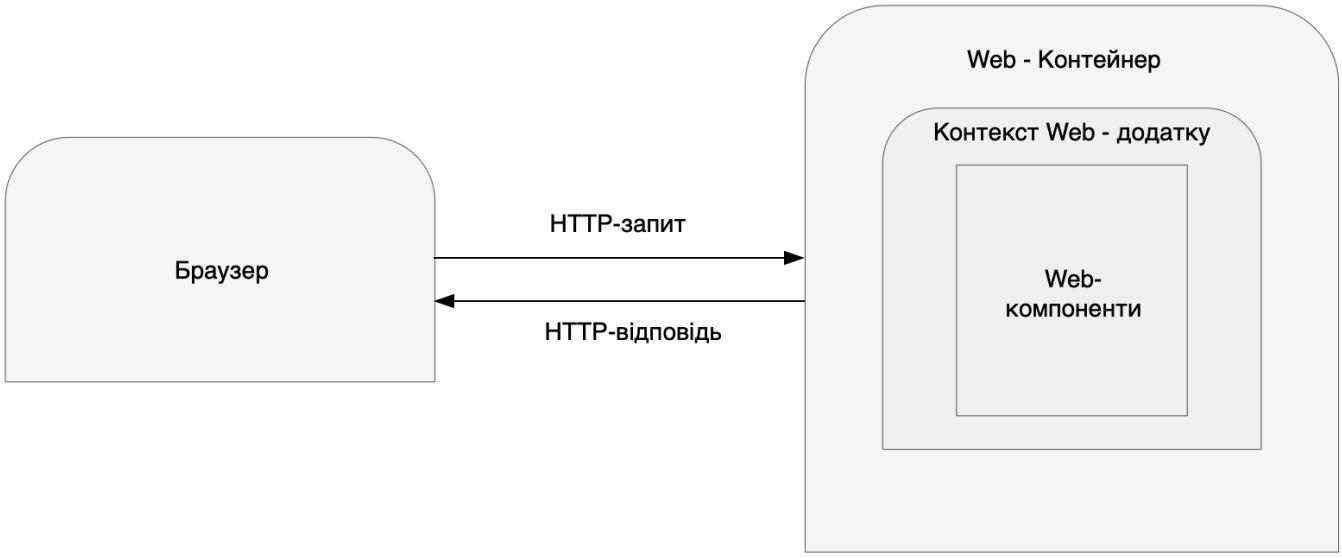


Рисунок 4.1 — Схема роботи web-додатку

4.3. Шаблон архітектури проекту

Даний програмний продукт був розроблений, використовуючи шаблон проектування MVC (Model – View – Controller). Приклад схеми роботи веб додатку на рисунку 3.1.

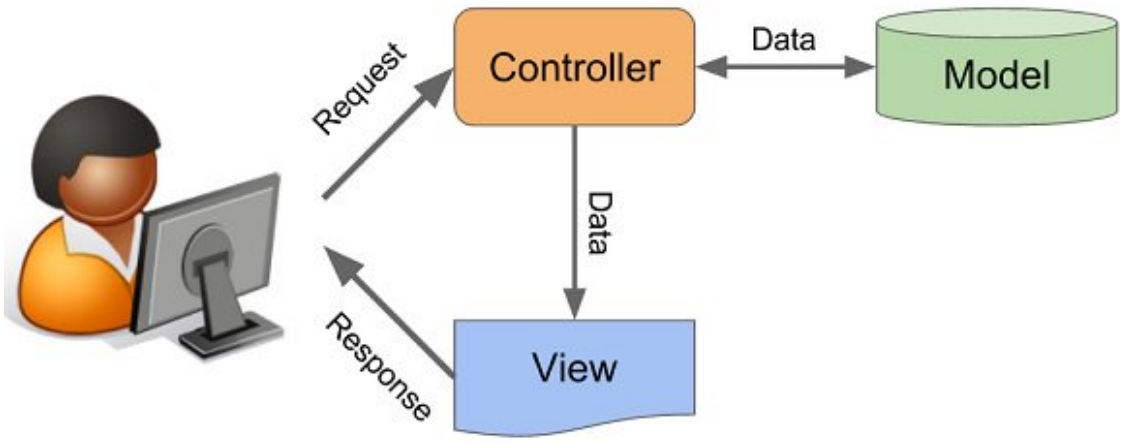


Рисунок 3.1 – Схема роботи веб додатку з використанням шаблону MVC

*Модель* представляє дані і не робить нічого іншого. Модель не залежить від контролера або виду.

*Перегляд* відображає дані моделі і посилає дії користувача (наприклад, натискання кнопок) на контролер. Перегляд може:

1. бути незалежними як від моделі, так і від контролера;
2. фактично бути контролером, і тому залежить від моделі.

*Контролер* надає модельні дані перегляду та інтерпретує дії користувача, такі як клацання кнопок. Контролер залежить від виду і моделі. У деяких випадках

контролер і перегляд є одним і тим же об'єктом.

Шаблон дизайну MVC вводить клас контролерів між поданням і моделлю для видалення залежностей перегляду моделі. З видаленими залежностями модель і, можливо, перегляд можна зробити багаторазовою без змін. Це робить реалізацію нових функцій та технічного обслуговування легким. Користувачі отримують стабільне програмне забезпечення швидко [19].

## 4.5 Архітектура програмного комплексу

Використовуючи шаблон MVC, ми спроектували архітектуру програми. Кожен елемент відповідає за певну реалізацію компонентів додатку (рисунок 3.2). Правильна архітектура забезпечує безперебійну роботу застосунку, а також дозволяє контролювати розробку.


Рисунок 3.2 – Архітектура програмного комплексу

В моделі знаходяться формули, за якими розраховуються ізотропні поверхні, коефіцієнти квадратичних форм, а також візуалізація поверхонь за допомогою бібліотеки Jzy3d. Контролер отримує дані з клієнту, перевіряє коректність вводу користувачем цих даних та відправляє результат для клієнта. Вигляд відповідає за клієнтський інтерфейс для вводу даних, а також за візуалізацію прикладу ізотропної поверхні.

## 4.6 Потенційні користувачі

У подальшому така система може бути використана для проектування споруд в архітектурі або в розробці медичного інструментарію, так як мінімальні поверхні мають оптимальний розподіл тиску по всій площі об’єкта, а можливість розробки додаткових модулів додає системі гнучкості.

## 4.7 Висновки до розділу

В даному розділі ми розглянули архітектуру створеного додатку, особливості інтерфейсу, а також схему роботи веб додатку.

Створений інтерфейс розроблений, враховуючи потреби користувача при користуванні веб додатками, що робить навігацію застосунком простішою, інтуїтивно зрозумілою.

Роботу із запитами і відповідями між клієнтом і сервером забезпечує контейнер, який містить в собі контекст веб додатку такі як модель – вид – контролер.

Розроблена архітектура, яка базувалась на шаблоні MVC, дозволила створити додаток, у якому всі компоненти працюють злагоджено.

# 5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 5.1 Опис web-додатку

Коли користувач запускає додаток, він потрапляє на головну сторінку (рисунок 5.1). Вона містить в собі інформацію про головну задачу, яку виконує сайт – побудова ізотропних поверхонь.

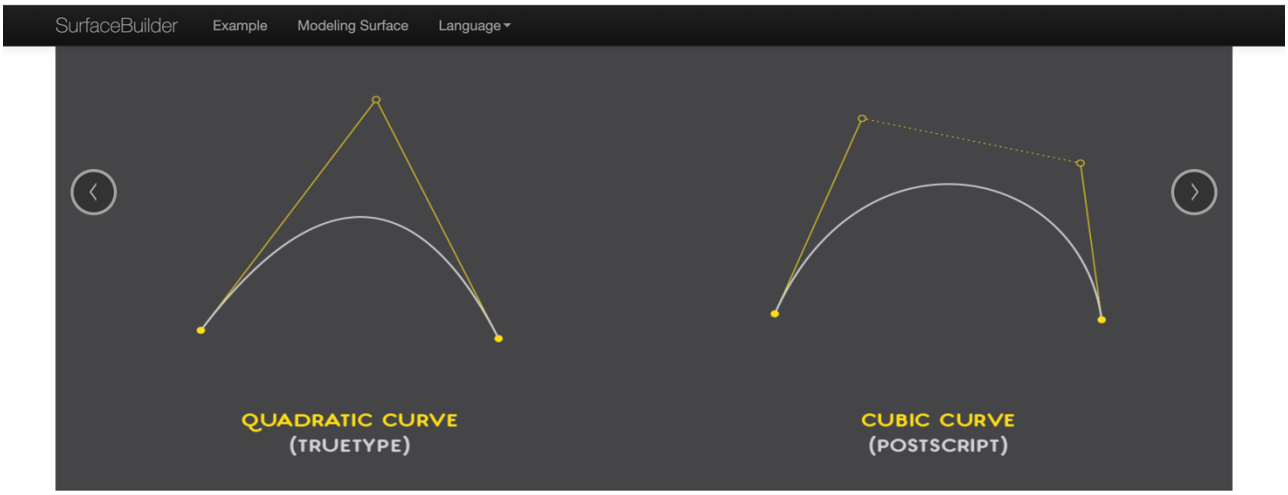


Рисунок 5.1 – Головна сторінка web-додатку

Користувач має змогу подивитись приклад побудови ізотропної поверхні (рисунок 5.2), яка побудована на основі кривої Без’є, натиснувши на кнопку “Example” головного меню.

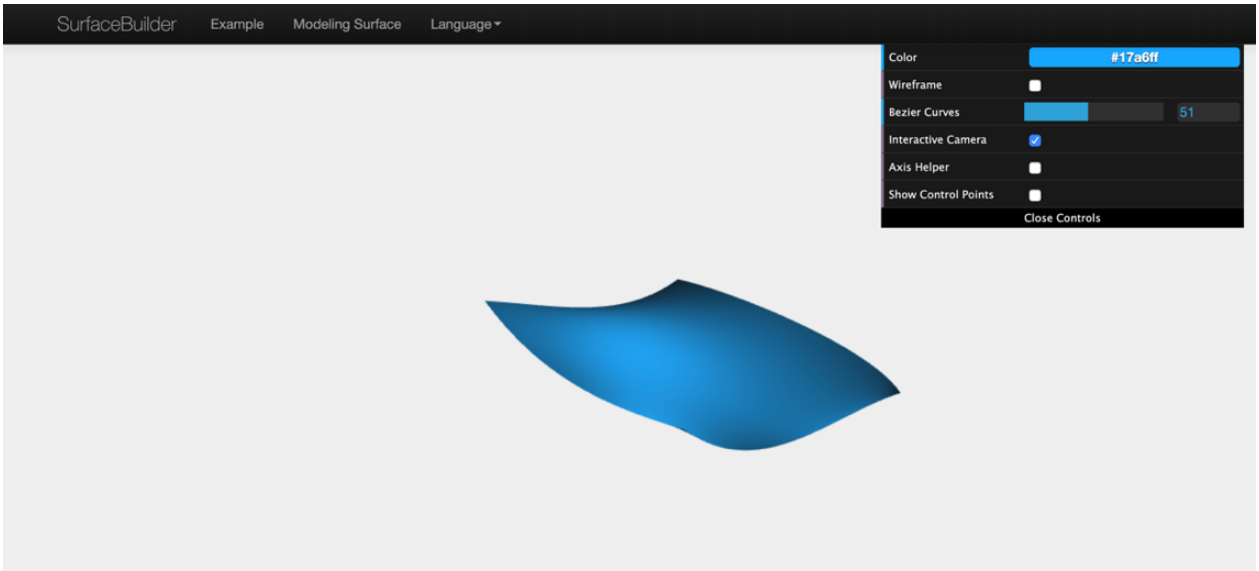


Рисунок 5.2 – Приклад побудови ізотропної поверхні

Тут користувач може змінювати колір поверхні, увімкнути опцію тріангуляції, змінити масштаб, розташування камери і джерела світла, переміщати контрольні точки, які будуть змінювати форму моделі.

Для того, щоб змоделювати поверхню, потрібно зайти в розділ «Modeling Surface» (рисунок 5.3).

Користувач повинен ввести коефіцієнти $a_0, a_1, a_2, a_3$ – ці коефіцієнти використовуються для розрахування реперних точок, що визначають умову ізотропності поверхні. Ці коефіцієнти комплексні числа виду $a + ib$, де $a -$ реальна частина, $b -$ уявна, тому користувач повинен ввести «Real» і «Imaginary» частини.

Натиснувши на кнопку “Create your surface” дані відправляються на сервер, оброблюються контейнером сервлета, перевіряються на коректність вводу. Введені значення підставляються у формулу по якій розраховується поверхня і з’являється вікно із побудованою поверхнею (рисунок 5.4)

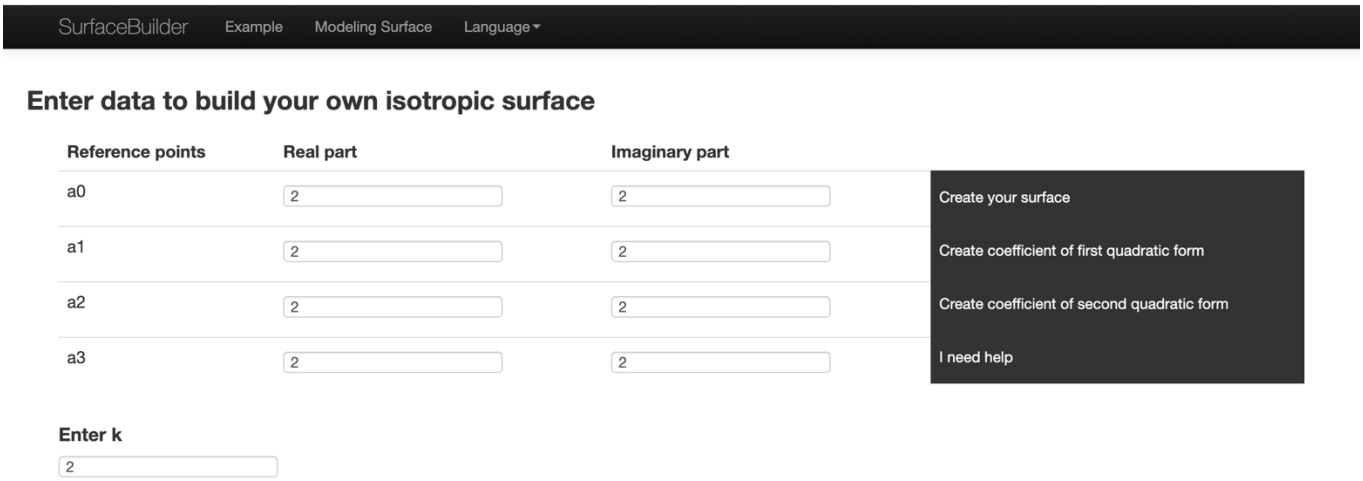


Рисунок 5.3 – Інтерфейс для вводу даних

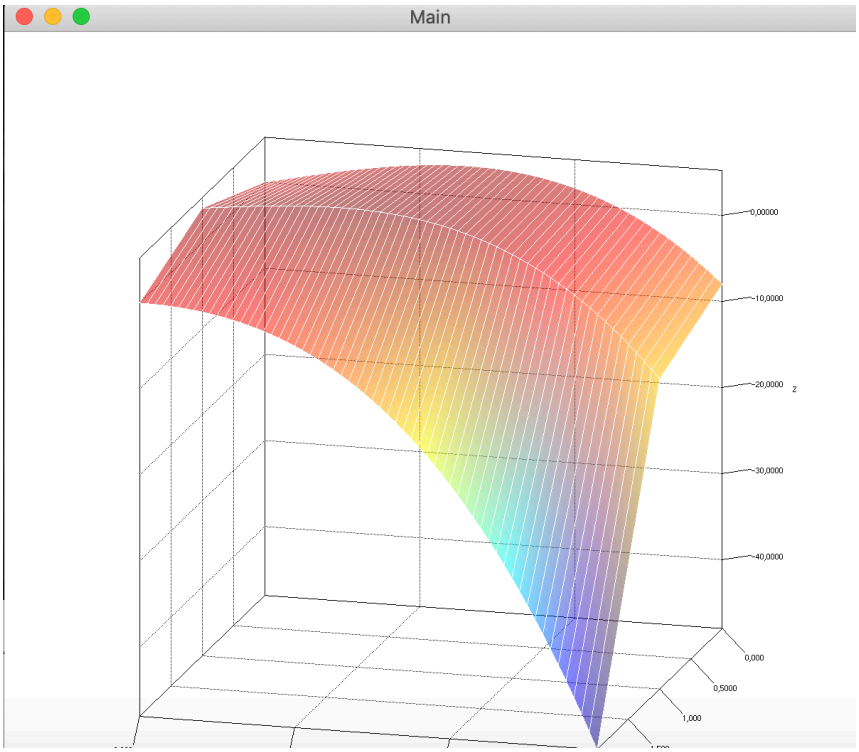


Рисунок 5.4 – Побудована ізотропна поверхня

Також є можливість побудувати коефіцієнти першої та другої квадратичних форм. Натиснувши на кнопку «Create coefficients of first quadratic form», користувач

отримає зображення $E$ (рисунок 5.5), $F$ (рисунок 5.6), $G$ (рисунок 5.7) – коефіцієнтів першої квадратичної форми. Якщо ж користувач натисне кнопку «Create coefficients of second quadratic form», з'явиться зображення $L$ (рисунок 5.8), $M$ (рисунок 5.9), $N$ (рисунок 4.10) – коефіцієнтів другої квадратичної форми.

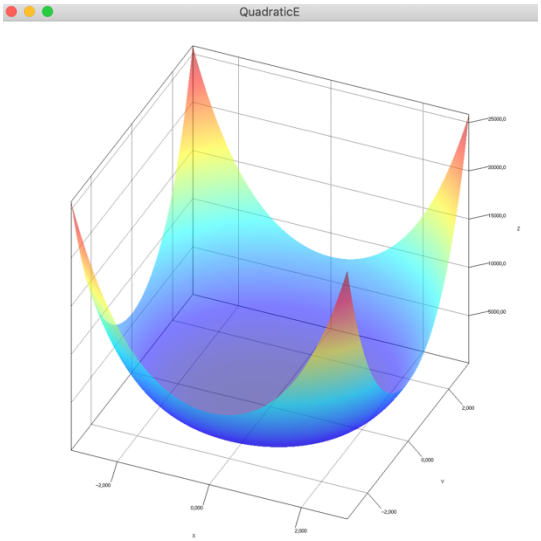


Рисунок 5.5 – Побудова коефіцієнта $E$

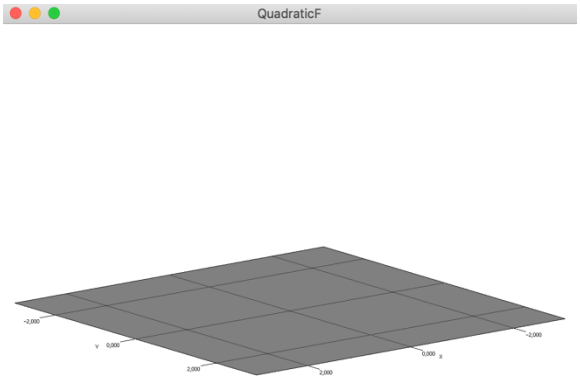


Рисунок 5.6 – Побудова коефіцієнта $F$

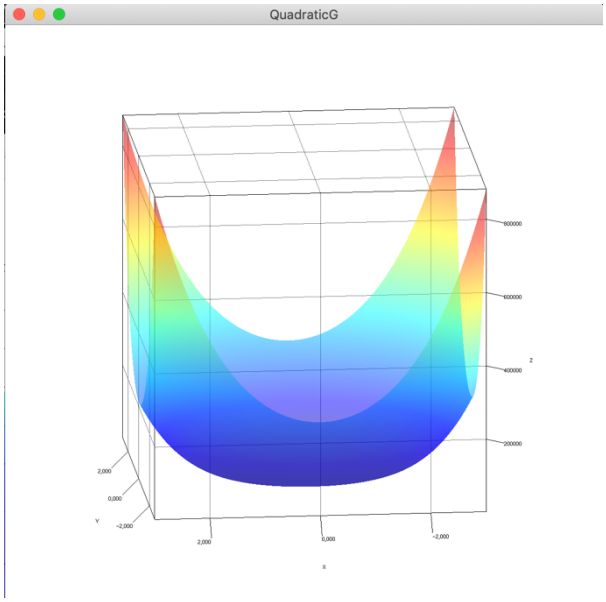


Рисунок 5.7 – Побудова коефіцієнта $G$

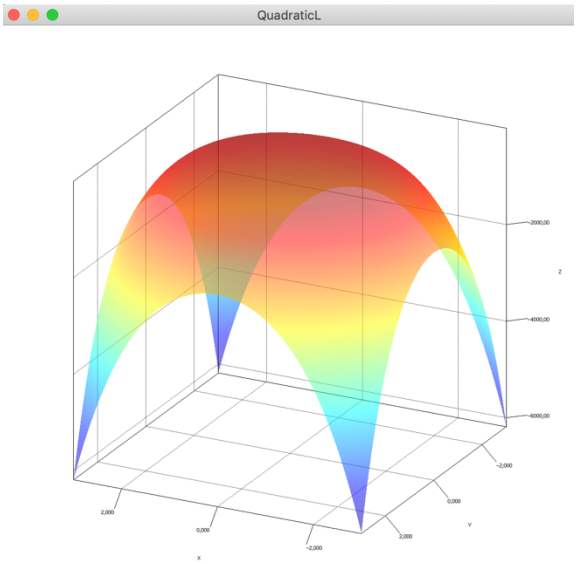


Рисунок 5.8 – Побудова коефіцієнта $L$

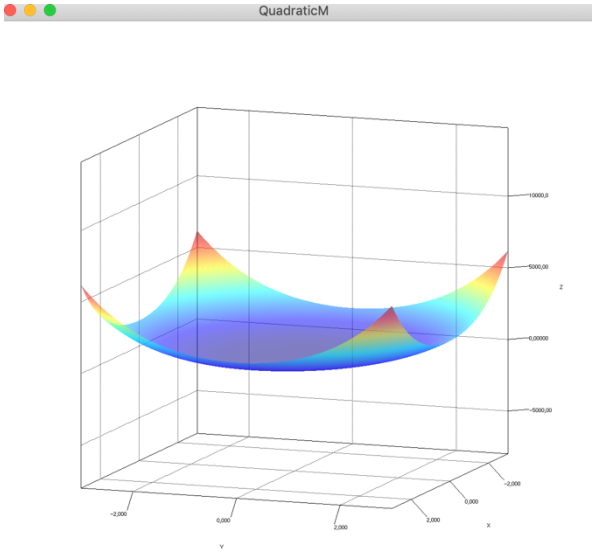


Рисунок 5.9 – Побудова коефіцієнта $M$

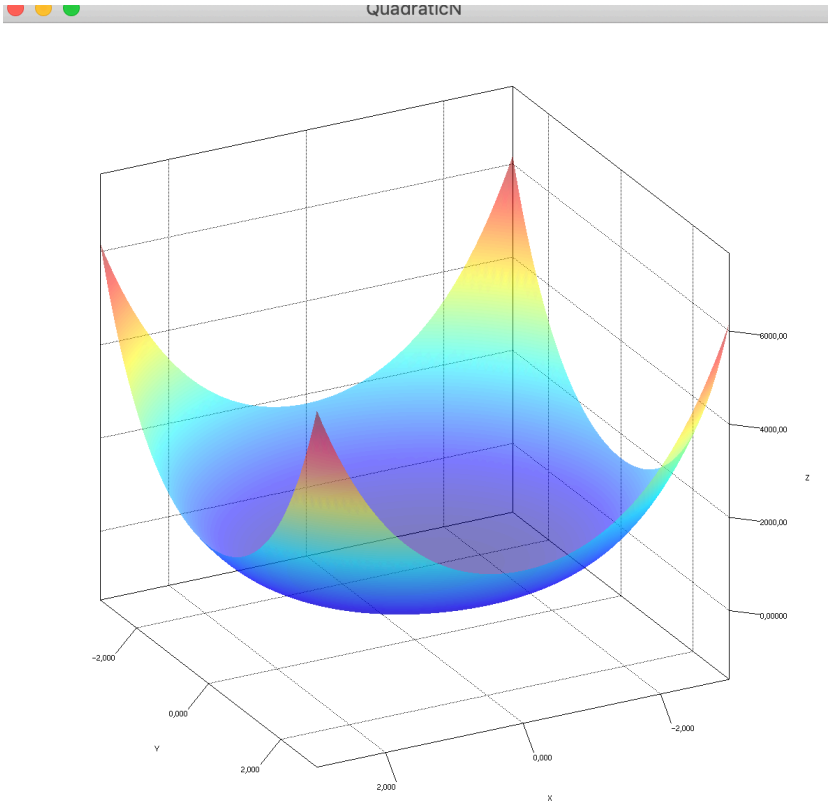


Рисунок 5.10 – Побудова коефіцієнта $N$

В додатку передбачений ввід замість чисел символів. Сервлет буде перенаправляти на цю ж сторінку, але із заголовком про помилку вводу, яка виникла (рисунок 5.11)

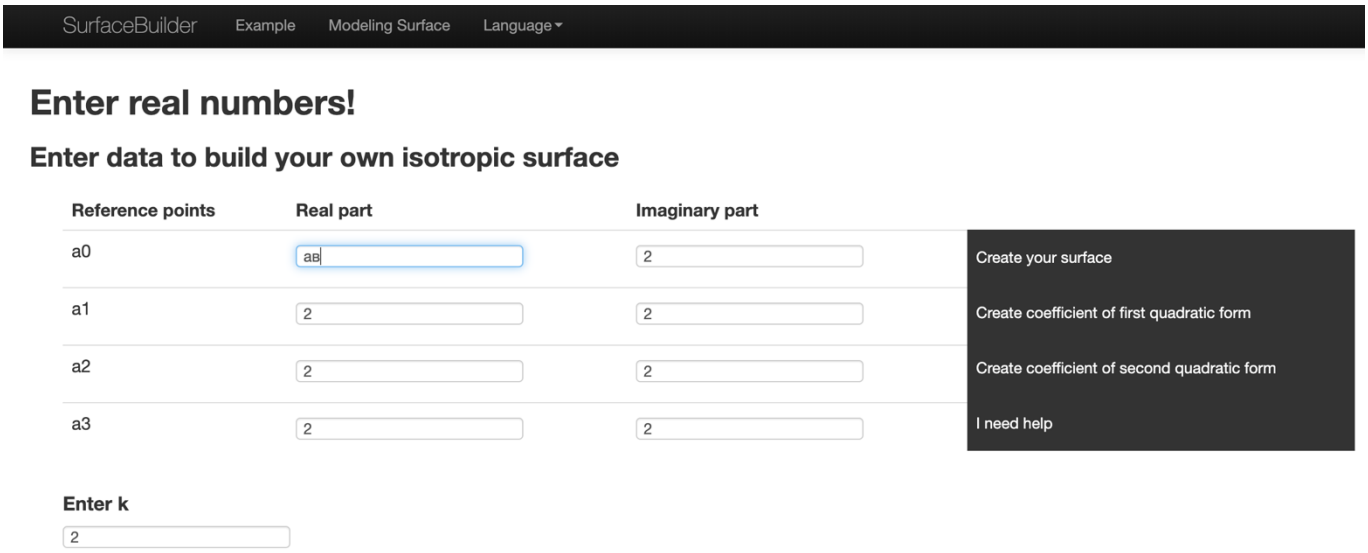


Рисунок 5.11 – Валідація вводу даних

Також є перевірка, чи немає пустих полів. Якщо ми не ввели значення в текстове поле, то з’явиться підказка про те, що потрібно заповнити це поле (рисунок 5.12).

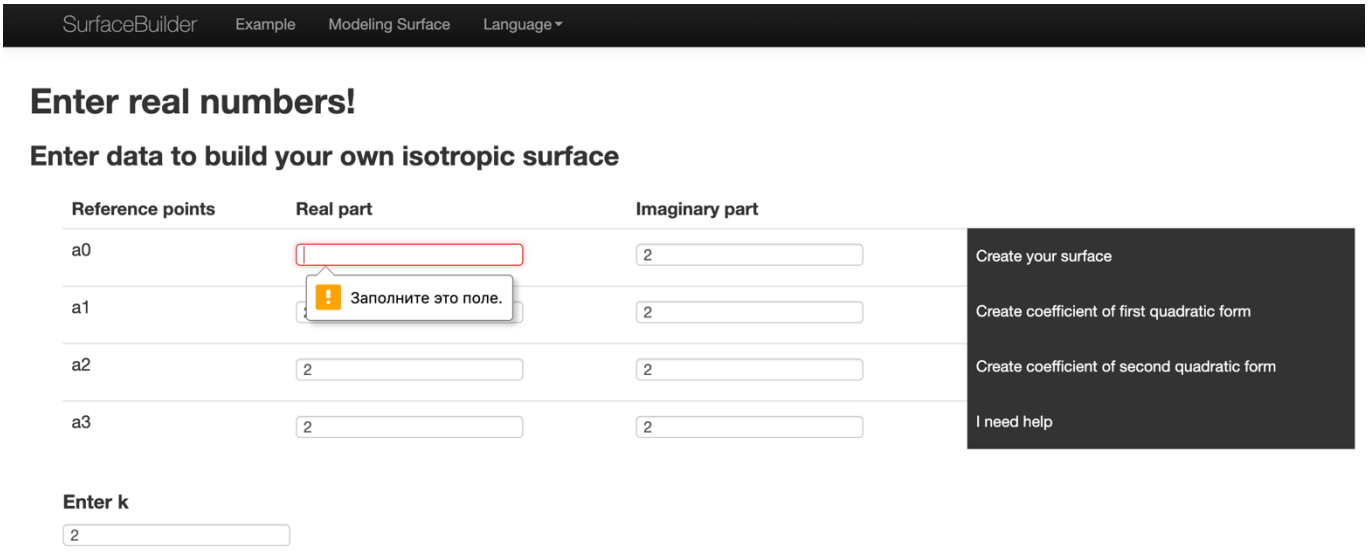


Рисунок 5.12 – Перевірка заповненості полів

Користувач також має змогу скористатись корисною інформацією від розробників, натиснувши кнопку «I need help» (рисунок 5.11).

- Необхідно ввести справжні числа;
- Після заповнення всіх текстових полів зачекайте хвилину, і вікно з вашою поверхнею з’явиться;
- Повернути: клацнути лівою кнопкою миші та перетягнути мишею;
- Масштаб: накрутіть колесо миші;
- Z Shift: клацніть правою кнопкою миші і перетягніть мишею;
- Анімація: двічі клацніть лівою кнопкою миші;
- Знімок екрана: натисніть 's'.

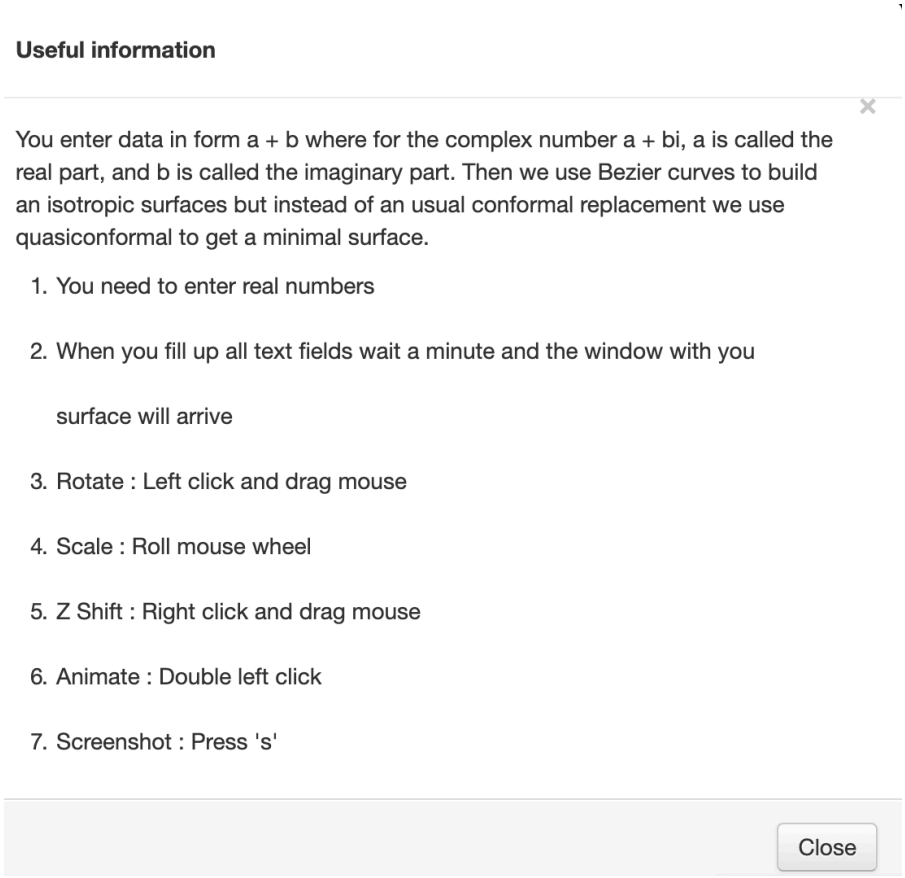


Рисунок 5.13 – Модальне вікно з корисною інформацією

Є також можливість зміни мови інтерфейсу. Кнопка верхнього меню «Language» відповідає за це (рисунок 5.14)

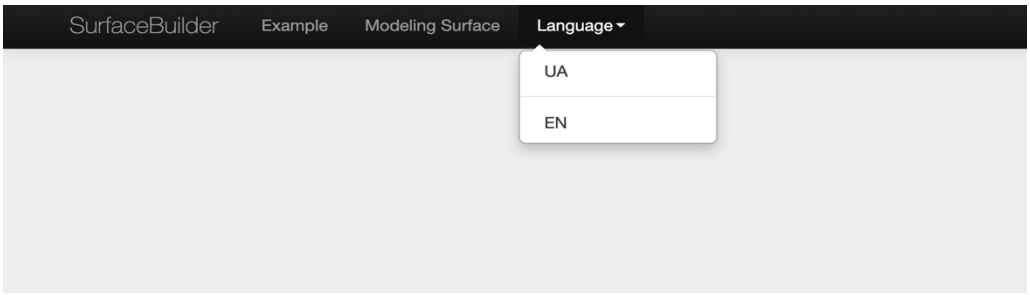


Рисунок 5.14 – Зміна мови інтерфейсу

## 5.2 Висновки до розділу

В даному розділі ми здійснили опис програмної реалізації, тобто кінцевий результат, який готовий до використання. Користувач матиме змогу переглянути приклади ізотропних поверхонь, побудувати ізотропні поверхні з квазіконформною заміною параметра, побудувати коефіцієнти першої квадратичної форми, другої квадратичної форми. При отриманні результатів можемо використати функцію анімації, масштабування або отримання знімку поверхонь для детального дослідження. Також ми представили варіанти проходженням валідації, яка була розроблена, щоб уникнути помилок та виключних ситуацій.

# ВИСНОВКИ

В результаті роботи на дипломної роботою було проведено аналіз моделювання ізотропних об’єктів, в результаті якого було виявлено, що ці поверхні мають специфічні властивості. В результаті досліджень було виявлено, що такі моделювання таких поверхонь необхідно вдосконалювати та застосовувати комп’ютерні технології ізотропної геометрії.

На основі аналізу предметної області та визначення мети роботи, були сформульовані завдання досліджень:

1. проаналізувати основні способи представлення поверхонь на основі ізотропних характеристик;
2. проаналізувати методи моделювання ізотропних поверхонь з квазіконформною заміною параметра;
3. удосконалити метод побудови ізотропних поверхонь з квазіконформною заміною параметра;
4. розробити алгоритм візуалізації ізотропних поверхонь з квазіконформною заміною параметра;
5. здійснити програмну реалізацію розроблених методів, а саме побудова власне ізотропних поверхонь, а також квадратичних форм.

На основі запропонованих способів моделювання та побудови поверхонь, був розроблений програмний додаток, який має такі функціональні можливості:

- побудова складних геометричних об’єктів на основі аналітичної функції кривої Без’є;
- побудова коефіцієнтів першої та другої квадратичних форм;
- масштабування поверхонь;

- анімація поверхонь;
- отримання знімку поверхні;
- зміна мови інтерфейсу;

На основі запропонованих алгоритмів моделювання та побудови поверхонь був розроблений програмний додаток, який по швидкодії перевершує існуючі аналоги. Створений програмний продукт значно спрощує процес моделювання ізотропних поверхонь з квазіконформною заміною параметра. Розроблений пограмний продукт може бути використаний, наприклад, в організаціях та установах, пов’язаних з архітектурною діяльністю.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Михайленко В. Е., Ковалев С. Н. Конструирование форм современных архитектурных сооружений / С. Н. Ковалев, В. Е. Михайленко – Киев: Будивельник, 1978. – 112 с.
2. Каган В. Ф., Основы теории поверхностей в тензорном изложении, ч. 1, М. — Л., 1947;
3. Курант Р., Роббинс Г., Что такое математика, пер. с англ., 2 изд., М., 1967
4. Бляшке В., Введение в дифференциальную геометрию, пер. с нем., М., 1957
5. Норден А. П. Теория поверхностей / А. П. Норден. – М.: ДМК Пресс, 1956. – 269 с.
6. Гуляев В.И. Расчёт оболочек сложной формы [Текст] / В.И. Гуляев, В.А. Баженов, Е.А. Гоцуляк, В.В. Гайдайчук. – К.: Будівельник, 1990. – 192 с.
7. Борисенко, О. А., Диференціальна геометрія і топологія : Навч. посібник для студ. — Харків : Основа, 1995 . - 304 с.
8. Аушева Н.М. Розробка узагальненого підходу щодо формування кривих та поверхонь дійсного простору на основі ізотропних характеристик / Н.М. Аушева // Журнал «Технологічний аудит та резерви виробництва». -No3/1(17).- Редакція «Східно-Європейського журналу передових технологій».-2014.-С.17-20.
9. Аушева Н.М. Моделювання поверхонь на основі квазіконформної заміни параметра / Н. М. Аушева, А. Л. Гурін // Сучасні проблеми моделювання. - 2017. - Вип. 10. - С. 17-21.
10. Binstock, Andrew (May 20, 2015). "Java's 20 Years of Innovation". Forbes. Archived from the original on March 14, 2016. Retrieved March 18, 2016.
11. JetBrains: What's New in IntelliJ IDEA [Electronic resource]. — Access mode: https://www.jetbrains.com/idea/whatsnew

12. Feick and Price, "The Market Maven: A Diffuser of Marketplace Information", Journal of Marketing, Jan 1987.
13. "Three.js API reference". Mrdoob.github.com. 2000-01-01. Retrieved 2013-05-09.
14. Louis Bavoil, Kevin Myers.Order Independent Transparency with Dual Depth Peeling. Retrieved February, 2008.
15. W3C Standerds HTML & CSS [Electronic resource]. — Access mode: https://www.w3.org/standards/webdesign/htmlcss
16. Otto, Mark (January 31, 2012). "Say hello to Bootstrap 2.0". Developer Blog. Twitter. Archived from the original on February 23, 2017. Retrieved February 23,2017.
17. Фисун А. П., Гращенко Л. А. и др. Теоретические и практические основы человеко-компьютерного взаимодействия: базовые понятия человеко-компьютерных систем в информатике и общественной безопасности / А.А. П. Фисун. - Деп. в ВИНИТИ 15.10. 2004 г. № 1624 - В 2004. - Орел: Орловский государственный университет, 2004. - 169 с. - (Рукопись).
18. James Duncan Davidson, Danny Coward (1999-12-17). Java Servlet Specification ("Specification") Version: 2.2 Final Release. Sun Microsystems pp. 43-46. Retrieved 27/07/2008.
19. Davis Ian “What are the Benefits of MVC?’ Internet Alchemy. Retrieved 29/11/2013.

Додаток 1

# Моделювання ізотропних поверхонь з квазіконформною заміною параметра

Специфікація

УКР.НТУУ“КПІ”.ТР4174_18Б

Аркушів 2

2019

| Позначення | Найменування | Примітки |
|---|---|---|
| Документація | | |
| УКР.НТУУ«КПІ ім. Ігоря Сікорського».ТР5174_18Б 81-1 | Записка | Пояснювальна записка |
| Компоненти | | |
| УКР.НТУУ«КПІ».ТР5174_18Б 12-1 | Текст програмного модулю | |
| УКР.НТУУ«КПІ».ТР5174_18Б 13-1 | Опис програми | |

# Додаток 2

## Моделювання ізотропних поверхонь з квазіконформною заміною параметра

Текст програмного модулю

УКР.НТУУ“КПІ”.ТР5174_18Б 12-1

Аркушів 8

2019

```
package controller.command;

import model.*;
import org.jzy3d.analysis.AnalysisLauncher;

import javax.servlet.http.HttpServletRequest;
import java.util.HashMap;

public class Bezier implements Command {
    private HashMap<String, Double> dots = new HashMap<>();

    @Override
    public String execute(HttpServletRequest request) throws Exception {
        try {
            dots.put("a0", Double.parseDouble(request.getParameter("a0")));
            dots.put("a1", Double.parseDouble(request.getParameter("a1")));
            dots.put("a2", Double.parseDouble(request.getParameter("a2")));
            dots.put("a3", Double.parseDouble(request.getParameter("a3")));
            dots.put("a0im", Double.parseDouble(request.getParameter("a0im")));
            dots.put("a1im", Double.parseDouble(request.getParameter("a1im")));
            dots.put("a2im", Double.parseDouble(request.getParameter("a2im")));
            dots.put("a3im", Double.parseDouble(request.getParameter("a3im")));
            dots.put("k", Double.parseDouble(request.getParameter("k")));
        }
        catch (NumberFormatException e){
            return "/indexWithEx.jsp";
        }
        switch (request.getParameter("ok")) {
            case "Create your surface":
                AnalysisLauncher.open(new Main(dots));
                break;
            case "Create coefficient of first quadratic form":
                AnalysisLauncher.open(new QuadraticE(dots));
                AnalysisLauncher.open(new QuadraticG(dots));
                AnalysisLauncher.open(new QuadraticF());
                break;
            case "Create coefficient of second quadratic form":
                AnalysisLauncher.open(new QuadraticM());
                AnalysisLauncher.open(new QuadraticN());
                AnalysisLauncher.open(new QuadraticL());
                break;
        }

        return "/index.jsp";
    }


}

package controller;

import controller.command.*;
import org.jzy3d.analysis.AnalysisLauncher;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.HashMap;
import java.util.Optional;

public class Servlet extends HttpServlet {

    private HashMap<String, Command> commands = new HashMap<String, Command>();


    @Override
    public void init() throws ServletException {
        commands.put("graphic", new Bezier());
        commands.put("example", new Example());
        commands.put("home", new Home());
        commands.put("modeling", new Modeling());
        commands.put("koef", new Bezier());

    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        try {
            processRequest(req, resp);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {

        try {
            processRequest(req, resp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void processRequest(HttpServletRequest req, HttpServletResponse resp)
            throws Exception {
        String path = req.getRequestURI();
        path = path.replaceAll(".*/app/", "");
        String page = getPage(path, req);
        if (page.contains("redirect")) {
            resp.sendRedirect(page.replace("redirect:", ""));

        } else {
            req.getRequestDispatcher(page).forward(req, resp);
        }
    }

    private String getPage(String path, HttpServletRequest req) throws Exception {
        String result = req.getContextPath();
```

```
        Optional<Command> command = Optional.ofNullable(commands.get(path));
        if (command.isPresent()) {
            result = command.get().execute(req);
        }
        return result;
    }

}
 package model;

import org.jzy3d.analysis.AbstractAnalysis;
import org.jzy3d.chart.factories.AWTChartComponentFactory;
import org.jzy3d.colors.Color;
import org.jzy3d.colors.ColorMapper;
import org.jzy3d.colors.colormaps.ColorMapRainbow;
import org.jzy3d.maths.Range;
import org.jzy3d.plot3d.builder.Builder;
import org.jzy3d.plot3d.builder.Mapper;
import org.jzy3d.plot3d.builder.concrete.OrthonormalGrid;
import org.jzy3d.plot3d.primitives.Shape;
import org.jzy3d.plot3d.rendering.canvas.Quality;

public abstract class AbstractQuadratic extends AbstractAnalysis {

    @Override
    public void init() throws Exception {
        Mapper mapper = new Mapper() {
            @Override
            public double f(double x, double y) {
                return -
18*(Math.pow(x,4)+2*Math.pow(y,2)*Math.pow(x,2)+Math.pow(y,4)+2*Math.pow(y,2)+1);
            }
        };

        // Define range and precision for the function to plot
        Range range = new Range(-3, 3);
        int steps = 80;

        // Create the object to represent the function over the given range.
        final Shape surface = Builder.buildOrthonormal(new OrthonormalGrid(range,
steps, range, steps), mapper);
        surface.setColorMapper(new ColorMapper(new ColorMapRainbow(),
surface.getBounds().getZmin(), surface.getBounds().getZmax(), new Color(1, 1, 1,
.5f)));
        surface.setFaceDisplayed(true);
        surface.setWireframeDisplayed(false);

        // Create a chart
        chart = AWTChartComponentFactory.chart(Quality.Advanced, getCanvasType());
        chart.getScene().getGraph().add(surface);
    }


}
 package model;

import org.apache.commons.math3.complex.Complex;
import org.nd4j.linalg.factory.Nd4j;

import java.util.HashMap;
```

```
 class Constant {

    private HashMap<String, Double> reader;
    double[] u = Nd4j.linspace(0, 1, 60).toDoubleVector();
    double[] v = Nd4j.linspace(0, 1, 60).toDoubleVector();
    double[] x = new double[60];
    double[] y = new double[60];
    double[] z = new double[60];
    double[] e = new double[60];
    double[] g = new double[60];
    double k;
    Complex a0;
    Complex a1;
    Complex a2;
    Complex a3;
    Complex x0;
    Complex x1;
    Complex x2;
    Complex x3;
    Complex y0;
    Complex y1;
    Complex y2;
    Complex y3;
    Complex z0;
    Complex z1;
    Complex z2;
    Complex z3;



    Constant(HashMap<String, Double> reader) {
        this.reader = reader;
        k = reader.get("k");
        a0 = new Complex(reader.get("a0"), reader.get("a0im"));
        a1 = new Complex(reader.get("a1"), reader.get("a1im"));
        a2 = new Complex(reader.get("a2"), reader.get("a2im"));
        a3 = new Complex(reader.get("a3"), reader.get("a3im"));
        x0 = a0.subtract(a2);
        x1 = a0.subtract(a2.add(a3));
        x2 = a0.subtract(a2.add(a3.multiply(2)));
        x3 = a0.subtract(a2.add(a3.multiply(2)));
        y0 = a0.add(a2);
        y1 = a0.add(a2).add(a3);
        y2 = a0.add(a2).add(a3.multiply(2));
        y3 = a0.add(a2).add(a3.multiply(4));
        z0 = a1.multiply(-1);
        z1 = a1.multiply(-1);
        z2 = a3.subtract(a1);
        z3 = a3.multiply(3).add(a1);

    }

}
package model;

import org.apache.commons.math3.complex.Complex;
import org.jzy3d.analysis.AbstractAnalysis;
import org.jzy3d.chart.factories.AWTChartComponentFactory;
```

```
import org.jzy3d.colors.Color;
import org.jzy3d.colors.ColorMapper;
import org.jzy3d.colors.colormaps.ColorMapRainbow;
import org.jzy3d.maths.Coord3d;
import org.jzy3d.plot3d.primitives.Point;
import org.jzy3d.plot3d.primitives.Polygon;
import org.jzy3d.plot3d.primitives.Shape;
import org.jzy3d.plot3d.rendering.canvas.Quality;
import org.nd4j.linalg.factory.Nd4j;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;


public class Main extends AbstractAnalysis {
    private HashMap<String, Double> reader = new HashMap<>();

    public Main(HashMap<String, Double> reader) throws Exception {
        this.reader = reader;

    }

    @Override
    public void init() {
        Constant constant = new Constant(reader);

        for (int i = 0; i < constant.u.length; i++) {
            constant.x[i] = quasiconformalReplacement(constant.u[i], constant.v[i],
constant.k, constant.x0, constant.x1, constant.x2, constant.x3);
        }

        for (int i = 0; i < constant.u.length; i++) {
            constant.y[i] = quasiconformalReplacement(constant.u[i], constant.v[i],
constant.k, constant.y0, constant.y1, constant.y2, constant.y3);
        }

        for (int i = 0; i < constant.u.length; i++) {
            constant.z[i] = quasiconformalReplacement(constant.u[i], constant.v[i],
constant.k, constant.z0, constant.z1, constant.z2, constant.z3);
        }

        // Define range and precision for the function to plot
        double[][] distDataProp = new double[][]{constant.x, constant.y, constant.z};
        List<Polygon> polygons = new ArrayList<Polygon>();
        for (int i = 0; i < distDataProp.length - 1; i++) {
            for (int j = 0; j < distDataProp[i].length - 1; j++) {
                Polygon polygon = new Polygon();
                polygon.add(new Point(new Coord3d(i, j, distDataProp[i][j])));
                polygon.add(new Point(new Coord3d(i, j + 1, distDataProp[i][j +
1])));
                polygon.add(new Point(new Coord3d(i + 1, j + 1, distDataProp[i + 1][j
+ 1])));
                polygon.add(new Point(new Coord3d(i + 1, j, distDataProp[i +
1][j])));
                polygons.add(polygon);
            }
        }

        // Create the object to represent the function over the given range.
```

```
        Shape surface = new Shape(polygons);
        surface.setColorMapper(new ColorMapper(new ColorMapRainbow(),
surface.getBounds().getZmin(), surface.getBounds().getZmax(), new Color(1, 1, 1,
.5f)));
        surface.setFaceDisplayed(true);
        surface.setWireframeDisplayed(true);

        // Create a chart
        chart = AWTChartComponentFactory.chart(Quality.Advanced, getCanvasType());
        chart.getScene().getGraph().add(surface);
    }

    private double quasiconformalReplacement(double u, double v, double k,
                                             Complex r0, Complex r1, Complex r2,
Complex r3) {
        return r0.getReal() *
                (1 - 3 * u + 3 * Math.pow(u, 2) - 3 * Math.pow(v, 2) * Math.pow(k, 2)
- Math.pow(u, 3) + 3 * u * Math.pow(v, 2) * Math.pow(k, 2)) -
                r0.getImaginary() * (-3 * v * k + 6 * u * v * k - 3 * Math.pow(u, 2)
* v * k + Math.pow(v, 3) * Math.pow(k, 3)) - (-3 * r1.getReal() * (1 - 2 * u +
Math.pow(u, 2) - Math.pow(v, 2) * Math.pow(k, 2)) + 3 * r1.getImaginary() * (-2 * v *
k + 2 * u * k * v)) * u +
                (-3 * r1.getImaginary() * (1 - 2 * u - Math.pow(u, 2) - Math.pow(v,
2) * Math.pow(k, 2)) - 3 * r1.getReal() * (-2 * v * k + 2 * u * v * k)) * v * k -
                ((-3) * r2.getReal() * (1 - u) - 3 * r2.getImaginary() * v * k) *
(Math.pow(u, 2) - Math.pow(v, 2) * Math.pow(k, 2)) + 2 * (-3 * r2.getImaginary() * (1
- u) + 3 * r2.getReal() * v * k) * u * v * k +
                r3.getReal() * (Math.pow(u, 3) - 3 * u * Math.pow(v, 2) * Math.pow(k,
2)) - r3.getImaginary() * (3 * Math.pow(u, 2) * v * k - Math.pow(v, 3) * Math.pow(k,
3));

    }


}
public class QuadraticE extends AbstractAnalysis {
    private HashMap<String, Double> reader = new HashMap<>();

    public QuadraticE(HashMap<String, Double> reader) throws Exception {
        this.reader = reader;

    }
    public QuadraticE(){}
    @Override

    public void init() throws Exception {
        Constant constant = new Constant(reader);

        for (int i = 0; i < constant.u.length; i++) {
            constant.e[i] =9*Math.pow(constant.a3.getReal(),
2)*(2*Math.pow(constant.u[i],2)*Math.pow(constant.v[i],2)*Math.pow(constant.k,2)+
                    Math.pow(constant.v[i],4)*Math.pow(constant.k,4)+2*
Math.pow(constant.v[i],2)*Math.pow(constant.k,2)+1+2*Math.pow(constant.u[i],2)+Math.p
ow(constant.u[i],4))+
                    9*Math.pow(constant.a3.getReal(), 2)*(Math.pow(constant.v[i],
4)*Math.pow(constant.k,4)+2*Math.pow(constant.v[i], 2) * Math.pow(constant.k,
2)+2*Math.pow(constant.v[i], 2) * Math.pow(constant.k,
2)*Math.pow(constant.u[i],2)+1+2*Math.pow(constant.u[i],2)+Math.pow(constant.u[i],
4));
```

```
            System.out.println(constant.e[i]);
        }
        Mapper mapper = new Mapper() {
            @Override
            public double f(double x, double y) {
                return
constant.e[4]*(Math.pow(x,4)+2*Math.pow(y,2)*Math.pow(x,2)+Math.pow(y,4)+2*Math.pow(y
,2)+1);
            }
        };

        // Define range and precision for the function to plot
        Range range = new Range(-3, 3);
        int steps = 80;

        // Create the object to represent the function over the given range.
        final Shape surface = Builder.buildOrthonormal(new OrthonormalGrid(range,
steps, range, steps), mapper);
        surface.setColorMapper(new ColorMapper(new ColorMapRainbow(),
surface.getBounds().getZmin(), surface.getBounds().getZmax(), new Color(1, 1, 1,
.5f)));
        surface.setFaceDisplayed(true);
        surface.setWireframeDisplayed(false);

        // Create a chart
        chart = AWTChartComponentFactory.chart(Quality.Advanced, getCanvasType());
        chart.getScene().getGraph().add(surface);
    }

}
```

Додаток 2

# Моделювання ізотропних поверхонь з квазіконформною заміною параметра

Опис програмного модулю

УКР.НТУУ“КПІ”.ТР5174_18Б 13-1

Аркушів 5

2019

# АНОТАЦІЯ

Метою роботи було створення системи керування складними геометричними об’єктами побудованими на основі ізотропної геометрії з використанням квазіконформної заміни. Програма забезпечує моделювання ізотропних поверхонь, коефіцієнтів першої та другої квадратичних форм на основі введених значень для розрахунку реперних точок. Розроблений програмний продукт може бути використаний, наприклад, в організаціях та установах, пов’язаних з архітектурною діяльністю.

ЗМІСТ

1. Відомості про програмний модуль ..... 4

1.1. Опис логічної структури ..... 4

1.2. Вхідні та вихідні дані ..... 5

2. Використовувані технічні засоби ..... 6

# 1 ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

Даний програмний модуль розроблено у середовищі IntelliJ IDEA, , використовуючи типізовану мову програмування Java, не типізовану мову програмування JavaScript, бібліотеку Three.js, бібліотеку Jzy3d та деякі додаткові бібліотеки.

Програма призначена для моделювання ізотропних поверхонь з квазіконформною заміною параметра.

## 1.1. Опис логічної структури

Було розроблено багатоплатформний програмний продукт, основною задачею якого є побудова ізотропних поверхонь. В комп'ютерній графіці у якості напрямних кривих найчастіше застосовують криві Без'є. В залежності на основі якої функції створюється ізотропна напрямна крива, отримаємо відображення мінімальної поверхні. Функціональні можливості додатку також включать в себе:

- зміна порядку кривої Без'є;
- зміна коефіцієнта k при заміні;
- побудову коефіцієнтів першої квадратичної функції;
- побудову коефіцієнтів другої квадратичної функції;
- розрахунок середньої кривизни поверхонь;
- перевірка, чи дійсно поверхня мінімальна;
- налаштування зовнішнього виду поверхні;
- налаштування масштабу(збільшення/зменшення зображення);

- підтримка анімації компонентів;

Розроблена програма має прямий механізм побудови складних ізотропних об’єктів. Завдяки цьому така система може бути використана в архітектурі для візуалізації траєкторії різьбового профілю, для деяких зводів архітектурних споруд або в розробці медичних приладів.

## 1.2. Вхідні та вихідні дані

Вхідними даними для системи є коефіцієнти, які потрібні для розрахунку реперних точок ізотропної поверхні.

Вихідними даними є візуалізація ізотропної поверхні з квазіконформною заміною параметра.

# 2 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було протестовано в браузері Safari 12.1 на персональному комп’ютері, який працює на базі процесору 2,6 GHz Intel Core i5 та має 8 Гб оперативної пам’яті. Розроблене програмне забезпечення є кросбраузереним та кросплатформенним, що дозволяє запускати його на комп’ютерах будь-якої потужності та в будь-яких сучасних браузерах.